

第七章 高级汇编语言技术

7.1 宏汇编

7.2 重复汇编

7.3 条件汇编

7.4 高级语言结构

本章目标

1. 掌握宏汇编

- 定义、调用、展开

2. 掌握重复汇编

- 读程序、写结果

3. 了解条件汇编

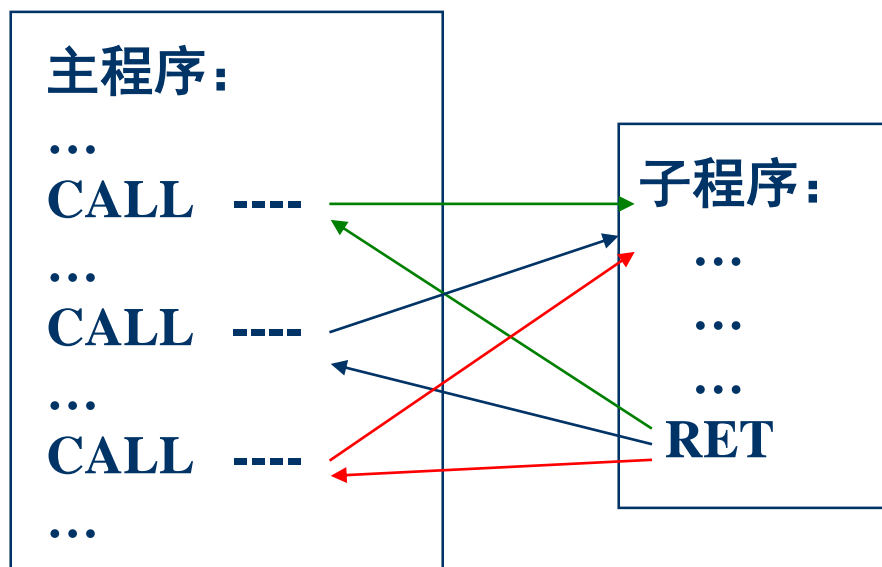
- 调用、展开

7.1 宏汇编

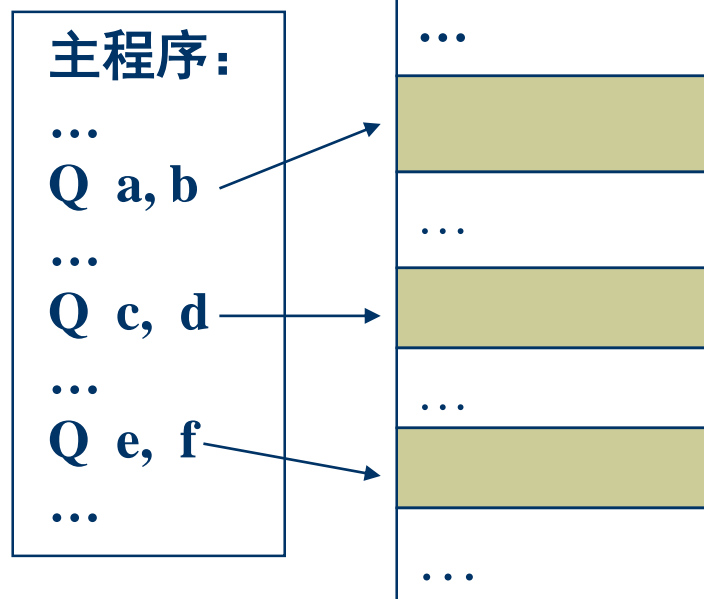
宏：源程序中一段有独立功能的程序代码。

宏指令：用户自定义的指令。在编程时，将多次使用的功能用一条宏指令来代替。

汇编语言程序 {
指令
伪指令（伪操作）
宏指令



优：模块化
省内存
缺：开销大



优：
参数传送
简单，执行效率高
缺：
占用内存
空间大

◆ 子程序

■ 优点

- 节省存储空间及程序设计所花的时间
- 提供模块化程序设计的条件
- 便于程序的调试及修改

■ 缺点

- 转子、返回，保存、恢复寄存器，参数的传送等，增加了程序的额外开销（操作所消耗的时间、占用的存储空间）

◆ 宏汇编的用途

- 当子程序本身较短或者需要传送的参数较多的情况下，使用宏汇编更加便利
- 为用户提供更加容易、更加灵活、更加向高级语言靠拢的汇编工具

```
multiply macro opr1,opr2,result
```

```
    push dx
```

```
    push ax
```

```
    mov ax,opr1
```

```
    imul opr2
```

```
    mov result,ax
```

```
    mov result+2,dx
```

```
    pop ax
```

```
    pop dx
```

```
endm
```

```
;
data segment
```

```
var dw ?
```

```
xyz dw ?,?
```

```
data ends
```

```
;
cseg segment
    assume cs:cseg,ds:data
start proc far
```

```
    mov ax,data
```

```
    mov ds,ax
```

```
    multiply cx,var,xyz[bx]
```

```
;
exit:    mov ax,4c00h
        int 21h
```

```
start endp
```

```
cseg ends
```

```
end start
```

源程序.ASM文件

```
multiply macro opr1,opr2,result
```

```
    push dx
```

```
    push ax
```

```
    mov ax,opr1
```

```
    imul opr2
```

```
    mov result,ax
```

```
    mov result+2,dx
```

```
    pop ax
```

```
    pop dx
```

```
endm
```

```
;
0000                                data segment
0000 ????                          var dw ?
0002 ???? ????                    xyz dw ?,?
0006                                data ends
```

```
;
0000                                cseg segment
                                assume cs:cseg,ds:data
0000                                start proc far
```

```
;
0000 B8 ---- R                    mov ax,data
0003 8E D8                        mov ds,ax
```

```
0005 52                          1 push dx
0006 50                          1 push ax
0007 8B C1                        1 mov ax,cx
0009 F7 2E 0000 R                1 imul var
000D 89 87 0002 R                1 mov xyz[bx],ax
0011 89 97 0004 R                1 mov xyz[bx]+2,dx
0015 58                          1 pop ax
0016 5A                          1 pop dx
```

```
;
0017 B8 4C00                      exit: mov ax,4c00h
001A CD 21                        int 21h
001C                                start endp
001C                                cseg ends
                                end start
```

汇编时展
开用一组
指令替代
宏指令

汇编后的.LIST文件

7.1.1 宏定义、宏调用和宏展开

宏定义：

`macro_name` **MACRO** [哑元表] ; 形参/虚参
[LOCAL 标号表]

.....
.....

 ; 宏定义体
ENDM

宏调用：（必须先定义后调用）

`macro_name` [实元表] ; 实参

宏展开：汇编程序把宏调用展开

宏定义体 \longrightarrow 复制到宏指令位置, 实参代虚参

LOCAL中的标号 \longrightarrow ??0000~??ffff

◆ 宏定义

■ 格式:

```
macro_name  MACRO  [哑元表]

               [ LOCAL    标号表 ]

               .....    (宏定义体)

               ENDM
```

```
multiply MACRO opr1,opr2,result
    mov ax,opr1
    imul opr2
    mov result,ax
    mov result+2,dx
ENDM
```

- * **MACRO**、**ENDM**是一对宏定义伪操作
- * 哑元表给出形式参数（虚参）
- * 宏定义体：一组有独立功能的程序段
- * 如果宏定义体有一个或多个**标号**，则必须用**LOCAL**伪操作列出所有的标号

宏定义

```
macro_name MACRO [哑元表]  
          [ LOCAL 标号表 ]  
          ..... (宏定义体)  
ENDM
```

- ◆ **宏调用：**定义了宏指令，就可以在程序中多次调用它

- **格式：**

macro_name [实元表] ; 实参

- * 实元表中的实元与哑元表中的哑元在位置上一一对应
- * 若实元数>哑元数，则多余的实元无效
- * 若实元数<哑元数，则多余的哑元作“空(NUL)”处理
- * 对宏指令的调用：**必须先定义后调用**
- * **实元：**可以是常数、寄存器、存储单元名、地址、表达式；也可以是操作码或操作码的一部分

汇编程序主要功能:

- 1、检查程序
- 2、测出源程序中的语法错误, 并给出错误信息
- 3、展开宏指令
- 4、产生源程序的机器语言目标程序, 并给出列表文件, 同时列出汇编语言和机器语言, XXX.LST

宏定义

```
macro_name MACRO [哑元表]  
[LOCAL 标号表]  
..... (宏定义体)  
ENDM
```

宏调用

```
macro_name [实元表]
```

◆ 宏展开:

- 源程序被汇编时, 汇编程序将对每个宏调用作宏展开
 - 用**宏定义体**替换宏指令名, 把**宏定义体**复制到调用宏指令的位置上, 同时用实元取代哑元
- 由LOCAL定义的标号也由 ??0000~??FFFF 偏移量替代 (其实质是自动给了一个新标号)
 - 多次宏调用展开时, 解决标号冲突问题

例7.1 两个16位的字操作数相乘

宏定义：（编程时）

```
multiply MACRO opr1, opr2, result
```

```
    push dx
```

```
    push ax
```

```
    mov ax, opr1
```

```
    imul opr2
```

```
    mov result, ax
```

```
    mov result+2, dx
```

```
    pop ax
```

```
    pop dx
```

```
ENDM
```

宏调用：（编程时）

```
multiply cx, var, xyz[bx]
```

宏展开：（汇编时）

```
1  push dx
1  push ax
1  mov ax, cx
1  imul var
1  mov xyz[bx], ax
1  mov xyz[bx]+2, dx
1  pop ax
1  pop dx
```

替代

1 表示这些指令由宏展开，同时也表示第一层展开结果，较早版本用 + 表示

7.1.2 宏定义中的参数

- **哑元：**实质上只是一个字符串构成的符号
- **实元：**可以是常数、寄存器、存储单元、地址、表达式；也可以是操作码或操作码的一部分
- 哑元和实元统称变元

例7.2 宏定义无变元

例7.3 变元是操作码

例7.4 变元是操作码的一部分

例7.6 变元是字符串

例7.7 变元是表达式

例7.2 保存寄存器

宏定义可以无变元

宏定义：

```
savereg  MACRO
    push  ax
    push  bx
    push  cx
    push  dx
    push  si
    push  di
ENDM
```

宏调用：

```
savereg
```

宏展开：

```
1  push  ax
1  push  bx
1  push  cx
1  push  dx
1  push  si
1  push  di
```

例7.3 变元可以是操作码

宏定义：


```
FOO MACRO P1, P2, P3
    MOV AX, P1
    P2 P3
ENDM
```

宏调用：

```
FOO WORD_VAR, INC, AX
```

宏展开：

```
1 MOV AX, WORD_VAR
1 INC AX
```



汇编程序汇编生成.OBJ文件时，生成这2条机器指令，并替代源程序中的宏调用指令

宏汇编操作符 & :: % : REQ ::=

◆ 符号1&符号2

- 文本替换操作符。宏展开时, 合并前后两个符号形成一个符号
- 符号可以是操作码、操作数或是一个字符串

例7.4 变元是操作码的一部分

宏定义: (源程序中定义)

```
leap macro cond, lab  
    j&cond lab  
endm
```

宏调用: (源程序中调用)

```
leap z, there  
.....  
leap nz, here
```

宏展开: (汇编时展开)

```
1  jz  there  
.....  
1  jnz here
```

◆ :: 注释

- 宏注释。宏展开时，若注释以一个分号开始，则该注释在宏扩展时出现。若注释以两个分号开始，则::后面的注释不予展开

例: Q **MACRO** m

 ; display a message

 ;; m is a string

ENDM

每次展开保留此注释

每次展开不保留此注释

◆ %表达式

- 表达式操作符。汇编程序将%后面的表达式立即求值转换为数字，并在展开时用这个数取代哑元，宏调用时使用

例7.7

宏定义：

```
MSG      MACRO  COUNT, STRING
MSG&COUNT DB STRING
ENDM

ERRMSG   MACRO  TEXT
CNTR=CNTR+1
MSG      %CNTR, TEXT
ENDM
```

宏调用：

```
.....
CNTR=0
ERRMSG   'SYNTAX ERROR'
.....
ERRMSG   'INVALID OPPERAND'
```

宏展开：

```
.....
CNTR=0
2  MSG1  DB  'SYNTAX ERROR'

.....
2  MSG2  DB  'INVALID OPPERAND'

.....
```

◆ : REQ

- 指定某个变元必须有。调用时必须有对应的实元，否则汇编时出错

例7.8: 宏定义

```
DIF  MACRO A, B
    DB B-A
ENDM
```

```
DIF1 MACRO A:REQ, B:REQ
    DB B-A
ENDM
```

```
DIF2 MACRO A:REQ, B
    DB B-A
ENDM
```

P267

◆ :=

- 为宏变元提供缺省值。

例7.9:

宏定义:

```
DIF3 MACRO A:=<10>,B:=<12>
    DB B-A
ENDM
```

宏调用:

```
DIF3
```

宏展开:

```
1 DB 12-10
```

宏调用:

```
DIF3 5, 8
```

宏展开:

```
1 DB 8-5
```

注意：宏指令名与指令助记符或伪操作名相同时，宏指令定义的优先级最高，即同名的助记符或伪操作名被汇编程序认为是宏指令。

例：

宏定义：

```
add    MACRO  opr1, opr2, result
.....
ENDM
```

宏调用：

```
.....
add    xx, yy, zz
purge  add                ; 取消宏定义
.....
```

建议宏指令名与指令助记符或伪操作名尽量不要相同

宏定义

```
macro_name MACRO [哑元表]  
          [ LOCAL 标号表 ]  
          ..... (宏定义体)  
ENDM
```

7.1.3 LOCAL伪操作

- ◆ 当在宏定义体中使用了标号，多次调用该宏定义时，则展开后会出现标号的多重定义，这是不允许的。
 - LOCAL伪操作可以解决这个问题
- ◆ LOCAL伪操作格式： **LOCAL 局部标号表**
 - 局部标号表中的每个符号，在汇编时每扩展一次便建立一个唯一的标号，形如??xxxx（xxxx的值在0000~FFFF之间），以保证汇编时生成符号名字的惟一性
 - LOCAL伪操作只能用在宏定义体内，必须是MACRO伪操作后的第一个语句，在MACRO和 LOCAL伪操作之间，不允许有注释和分号标志

例7.10 求绝对值(使用LOCAL伪操作)

宏定义:

absol

MACRO

oper

LOCAL

next

cmp oper, 0

jge next

neg oper

next:

ENDM

宏展开:

.....

1 **cmp var, 0**
1 **jge ??0000**
1 **neg var**
1 **??0000:**

.....

宏调用:

.....

absol

var

.....

absol

bx

.....

1 **cmp bx, 0**
1 **jge ??0001**
1 **neg bx**
1 **??0001:**

.....

例：定义延时程序的宏指令，在同一个程序中两次被调用的扩展情况

宏定义：

```
DELAY  MACRO
        LOCAL  LOP
        MOV    CX, 2801
LOP:    LOOP   LOP
        ENDM
```

宏调用：

```
DELAY
DELAY
```

汇编时宏扩展如下：

```
MOV    CX, 2801
1  ??0000:  LOOP  ??0000
MOV    CX, 2801
1  ??0001:  LOOP  ??0001
```

7.1.4 在宏定义内使用宏

- ◆ 宏指令嵌套有两种情况：
 - 宏定义体中含有宏调用
 - 必须先定义后调用
 - 宏定义体中含有宏定义

例7.12

宏定义:

```
INT21 MACRO FUNCTION
    MOV AH, FUNCTION
    INT 21H
ENDM

DISP MACRO CHAR
    MOV DL, CHAR
    INT21 02H
ENDM
```

宏调用:

DISP ‘?’

宏展开:

```
1  MOV      DL, ‘?’
2  MOV      AH, 02H
2  INT      21H
```


例7.13

宏定义:

```
DEFMAC MACRO MACNAM, OPERATOR
    MACNAM MACRO X, Y, Z
        PUSH AX
        MOV AX, X
        OPERATOR AX, Y
        MOV Z, AX
        POP AX
    ENDM
ENDM
```

用宏调用形成加法宏定义:

```
DEFMAC ADDITION, ADD
```

形成加法宏定义:

```
ADDITION MACRO X, Y, Z
    PUSH AX
    MOV AX, X
    ADD AX, Y
    MOV Z, AX
    POP AX
ENDM
```

宏调用:

```
ADDITION VAR1, VAR2, VAR3
```

宏展开:

```
1  PUSH AX
1  MOV  AX, VAR1
1  ADD  AX, VAR2
1  MOV  VAR3, AX
1  POP  AX
```

7.1.5 列表伪操作

(自学 *p271*)

列表伪操作：

.LALL：在LST清单中列出宏展开后的全部语句
(包括注释)。

.SALL：在LST清单中不列出任何宏展开后的语句。

.XALL：缺省的列表方式，只列出宏体中产生目标
代码的语句。

列表伪指令只影响列表文件，并不影响目标码的生成

7.1.6 宏库的建立与调用 (自学 P274)

建立宏库:

>EDIT MACRO.MAC

```
macro1 MACRO [哑元表]
```

```
.....  
ENDM
```

```
macro2 MACRO [哑元表]
```

```
.....  
ENDM
```

```
.....
```

```
macroN MACRO [哑元表]
```

```
.....  
ENDM
```

调用宏库:

>EDIT EXP.ASM

```
include MACRO.MAC
```

```
.....  
macro1 [实元表]
```

```
.....  
macro2 [实元表]
```

```
.....  
macroN [实元表]
```

```
.....
```

```
← purge macroN
```

7.1.7 删除宏定义

◆ 格式:

PURGE **macro_name** [, **macro_name**, ...]

- 删除不再使用的宏定义，使该宏定义为空
- 删除后，汇编程序再遇到该宏调用指令将忽略，也不会指示出错

7.2 重复汇编

用于连续产生完全相同或基本相同的一组代码。

重复伪操作 REPT

REPT 表达式

;重复块

ENDM

不定重复伪操作 IRP/IRPC

IRP 哑元, <自变量表>

;重复块

ENDM

IRPC 哑元, 字符串

;重复块

ENDM

7.2.1 重复伪操作

◆ 重复伪操作 REPT

格式： REPT 表达式
 ; 重复块
 ENDM

- 表达式：重复次数，结果应该是无符号常数
- 不一定要用在宏定义中，程序其他段中也可以用

例7. 15

```
X=0  
REPT 10  
X=X+1  
DB X  
ENDM
```

汇编展开后:

```
1 DB 1  
1 DB 2  
1 DB 3  
.....  
1 DB 10
```

例7.16 把字符 ‘A’到 ‘Z’ 的 ASCII 码填入数组TABLE

```
CHAR='A'  
TABLE LABEL BYTE  
      REPT 26  
      DB CHAR  
      CHAR=CHAR+1  
      ENDM
```

汇编展开后:

```
TABLE LABEL BYTE  
      1 DB 41H  
      1 DB 42H  
      1 DB 43H  
      .....  
      1 DB 5AH
```


7.2.2 不定重复伪操作

□ 不定重复伪操作 IRP / IRPC

1. IRP伪操作：用实际参量替换哑元

格式： IRP 哑元，〈自变量表〉
..... ； 重复块

ENDM

展开时，重复块中变量不定，无规律

- 每次重复用自变量表中的一项取代哑元，直到用完为止
- 重复次数由自变量的个数决定

例7. 20

```
IRP  REG, <AX,BX,CX,DX>  
    PUSH  REG  
ENDM
```

汇编展开后：

```
1  PUSH  AX  
1  PUSH  BX  
1  PUSH  CX  
1  PUSH  DX
```

例：在数据段产生字符区array，包括5个字符串 ‘NO.K’

```
data segment
array label byte
    IRP K, <1,2,3,4,5>
        db 'NO.&K'
    ENDM
data ends
```

汇编展开后

```
data segment
array label byte
1 db 'NO.1'
1 db 'NO.2'
1 db 'NO.3'
1 db 'NO.4'
1 db 'NO.5'
data ends
```

7.2.2 不定重复伪操作

2. IRPC伪操作：用字符串替换哑元

格式： IRPC 哑元，字符串
 ； 重复块
 ENDM

- 每次重复用字符串中的一个字符取代哑，直到用完为止
- 重复次数等于字符串中的字符数

例： 在数据段产生字符区array，包括5个字符串 ‘NO. K’

```
data segment
    array label byte
    IRPC K, 12345
    db 'NO.&K'
    ENDM
data ends
```

汇编展开后：





```
data segment
    array label byte
    1 db 'NO.1'
    1 db 'NO.2'
    1 db 'NO.3'
    1 db 'NO.4'
    1 db 'NO.5'
data ends
```

7.3 条件汇编

根据条件把一段源程序包括在汇编语言程序内或者排除在外。

一般格式：

IF ×× 自变量	; ××为条件
	; 自变量满足条件则汇编此块
[ELSE]	
	; 自变量不满足条件则汇编此块
ENDIF	

条件伪操作：

{	IF	表达式	;表达式$\neq 0$，则汇编
	IFE	表达式	;表达式$= 0$，则汇编
{	IF1		;在第一遍扫视期间满足条件
	IF2		;在第二遍扫视期间满足条件
{	IFDEF	符号	;符号已定义，则汇编
	IFNDEF	符号	;符号未定义，则汇编
{	IFB	<自变量>	;自变量为空，则汇编
	IFNB	<自变量>	;自变量不为空，则汇编
{	IFIDN	<字符串1>,<字符串2>	;串1与串2相同
	IFDIF	<字符串1>,<字符串2>	;串1与串2不同

例7.24 求3个变元中最大值放入AX，且变元 数不同时产生不同的程序段

宏展开:

宏定义:

```
MAX MACRO K,A,B,C
    LOCAL NEXT,OUT
    MOV AX,A
    IF K-1 ; 如果k-1≠0, 条件为真
    IF K-2
    {
        CMP C,AX
        JLE NEXT
        MOV AX,C
    }
    ENDIF
    NEXT: CMP B,AX
    JLE OUT
    MOV AX,B
    ENDIF
    OUT:
ENDM
```

宏调用:

```
MAX 1,P
MAX 2,P,Q
MAX 3,P,Q,R
```

1 MOV AX,P
1 ??0001:

1 MOV AX,P
1 ??0002: CMP Q,AX
1 JLE ??0003
1 MOV AX,Q
1 ??0003:

1 MOV AX,P
1 CMP R,AX
1 JLE ??0004
1 MOV AX,R
1 ??0004: CMP Q,AX
1 JLE ??0005
1 MOV AX,Q
1 ??0005:


```

MAX MACRO K, A, B, C
    LOCAL NEXT, OUT
    MOV AX, A
    IF K-1 ;如果k-1≠0, 条件为真
        IF K-2
            CMP C, AX
            JLE NEXT
            MOV AX, C
        ENDIF
    NEXT:
        CMP B, AX
        JLE OUT
        MOV AX, B
    ENDIF
OUT:
ENDM

;
data segment
P dw ?
Q dw ?
R dw ?
data ends
;
cseg segment
assume cs:cseg,ds:data
start proc far
;
    mov ax, data
    mov ds, ax
;
    MAX 1, P
    MAX 2, P, Q
    MAX 3, P, Q, R
;
exit:
    mov ax, 4c00h
    int 21h
start endp
cseg ends
end start

```

```

MAX MACRO K, A, B, C
    LOCAL NEXT, OUT
    MOV AX, A
    IF K-1 ;如果k-1≠0, 条件为真
        IF K-2
            CMP C, AX
            JLE NEXT
            MOV AX, C
        ENDIF
NEXT:
    CMP B, AX
    JLE OUT
    MOV AX, B
    ENDIF
OUT:
ENDM

;
0000 data segment
0000 P dw ?
0002 Q dw ?
0004 R dw ?
0006 data ends
;
0000 cseg segment
    assume cs:cseg,ds:data
    start proc far
;
0000 B8 ---- R mov ax, data
0003 8E D8 mov ds, ax
;
0005 A1 0000 R 1 MAX 1, P
0008 1 ??0001: MOV AX, P
;
0008 A1 0000 R 1 MAX 2, P, Q
000B 39 06 0002 R 1 MOV AX, P
000F 7E 03 1 CMP Q, AX
0011 A1 0002 R 1 JLE ??0003
0014 1 ??0003: MOV AX, Q
;
0014 A1 0000 R 1 MAX 3, P, Q, R
0017 39 06 0004 R 1 MOV AX, P
001B 7E 03 1 CMP R, AX
001D A1 0004 R 1 JLE ??0004
0020 39 06 0002 R 1 MOV AX, R
0024 7E 03 1 CMP Q, AX
0026 A1 0002 R 1 JLE ??0005
0029 1 ??0005: MOV AX, Q
;
0029 B8 4C 00 exit: mov ax, 4c00h
002C CD 21 int 21h
002E start endp
002E cseg ends
end start

```

例7. 25 根据跳转距离生成不同跳转指令

宏定义:

```
BRANCH  MACRO  X
    IF    ($-X) LT 128
        JMP SHORT X
    ELSE
        JMP NEAR PTR X
    ENDIF
ENDM
```

宏调用:

```
BRANCH AA
```

宏展开:

如果相对于AA距离小于
128, 宏展开

```
1  JMP SHORT AA
```

否则产生

```
1  JMP NEAR PTR AA
```

例7.26 在宏定义的递归调用中，使用条件伪操作结束宏递归

X和 2^N 相乘，即X左移N次

宏定义：

```
POWER MACRO X, N
    SAL X, 1
    COUNT=COUNT+1
    IF COUNT-N
    POWER X, N
    ENDIF
ENDM
```

宏调用：

```
COUNT=0
POWER AX, 3
```

宏展开：

```
1  SAL AX, 1
2  SAL AX, 1
3  SAL AX, 1
```

例 7.28 (p285):

divide **macro** dividend,divisor,quotient

local comp, out

cnt=0

ifndef dividend

cnt=1

endif

ifndef divisor

cnt=1

endif

ifndef quotient

cnt=1

endif

if cnt

exitm

endif

mov ax, dividend

mov bx, divisor

sub cx, cx

comp:

cmp ax, bx

jb out

sub ax, bx

inc cx

jmp comp

out:

mov quotient, cx

endm

7.4 高级语言结构 (自学 *p293*)

- ◆ MASM 6.0引入了几种更接近高级语言编程的高级语言结构，如以下标准宏指令
 - .IF/.ELSEIF/.ELSE/.ENDIF
 - .WHILE/.ENDW
 - .REPEAT/.UNTIL
 - .REPEAT/.UNTILECXZ
 - .BREAK
 - .CONTINUE
- ◆ 汇编程序将按标准指令段展开，形成一串指令完成特定操作

. IF/. ELSEIF/. ELSE/. ENDIF

格式:

```
. IF expression1  
  (汇编语言语句组1)  
. ELSEIF expression2  
  (汇编语言语句组2)  
. ELSEIF expression3  
  (汇编语言语句组3)  
  ⋮  
. ELSE  
  (汇编语言语句组n)  
. ENDIF
```

.IF宏指令在汇编时会产生比较
(**CMP**) 和条件跳转两条指令

汇编时这里产生一条**JMP**指令

```
.IF AL==“A”  
CALL DISP  
.ENDIF
```

```
CMP AL, “A”  
JNZ  NOTA  
CALL DISP
```

高级语言结构中使用的表达式

◆ 表达式中的操作符

机器指令中有对应的操作

- == 相等
- != 不等
- > 大于
- >= 大于或等于
- < 小于
- <= 小于或等于
- & 位测试
- ! 逻辑非
- && 逻辑与
- || 逻辑或

◆ 表达式格式

1) 测试条件码的值 (=1时表达式值为真)

- ZERO?
- CARRY?
- OVERFLOW?
- SIGN?
- PARITY?

机器指令中有对应的条件转移

2) 操作数构成的表达式

- reg op reg
- reg op memory
- reg op constant
- memory op constant

机器指令中有对应的操作数表示形式

.IF reg == constant



CMP reg, constant
JNZ xxxxx

- 高级语言语句在汇编时汇编程序将生成标准的机器指令串替代高级语言语句

更加易于编程和可读性

```
.IF AL=="A"  
    CALL DISP  
.ELSEIF AL=="B"  
    CALL DISP  
.ELSE  
    MOV AL, "N"  
    CALL DISP  
.ENDIF
```

汇编展开

```
CMP AL, "A"  
JNZ NOTA  
CALL DISP  
JMP DONE  
NOTA: CMP AL, "B"  
JNZ NOTB  
CALL DISP  
JMP DONE  
NOTB: MOV AL, "N"  
CALL DISP  
DONE:
```

例 7.32 (p294)

练习举例

P301 7.1

7.1 定义宏指令CLRB，完成用空格符将一字符区中的字符取代的工作。字符区的首地址及其长度为变元。（两种实现方案）

```
CRLB1  MACRO  ADDR, CUNT
        LOCAL  NEXT
        MOV    SI, OFFSET  ADDR
        MOV    AL, 20H  ;
        ' , ' =20H
        MOV    CX, CUNT
NEXT:   MOV    [SI], AL
        INC    SI
        LOOP   NEXT
        ENDM
```

```
CRLB2  MACRO  ADDR, CUNT
        LEA    DI, ADDR
        MOV    AL, ' '
        CLD                                ; DF=0
        MOV    CX, CUNT
        REP    STOSB
        ENDM
```

作业

7.3 (1)

7.4

(源字符串首址 = `arrys`,
目的字符串首址 = `arryd`)

7.7

7.8

7.11